IET Renewable Power Generation

The Institution of Engineering and Technology

Cluster Analysis of Wind Turbine Alarms for Characterising and Classifying Stoppages

ISSN 1751-8644
doi: 0000000000
www.ietdl.org

Kevin Leahy[1] ✉, *Colm Gallagher*[1], *Peter O'Donovan*[1], *Dominic T.J. O'Sullivan*[1]

[1] *Department of Civil & Environmental Engineering, University College Cork, Cork, Ireland*
✉ *E-mail: kevin.leahy@umail.ucc.ie*

Abstract: Turbine alarm systems can give useful information to remote technicians on the cause of a fault or stoppage. However, alarms are generally generated at much too high a rate to gain any meaningful insight from on their own, so generally require extensive domain knowledge of the specific alarm system. By grouping together commonly occurring alarm sequences, the burden of analysis can be reduced. Instead of analysing many individual alarms that occur during a stoppage, the stoppage can be linked to a commonly occurring sequence of alarms and that sequence's associated characteristics.
In this research, we present a methodology to identify relevant alarms from specific turbine assemblies and group together similar alarm sequences as they appear during stoppages. Batches of sequences associated with 456 different stoppages are created, and features are extracted from these batches representing the order the alarms appeared in. The batches are then grouped together using clustering techniques, and evaluated using silhouette analysis and manual inspection. The results show that almost half of all stoppages can be attributed to one of 15 different alarm sequences. When one of these alarm sequences appears, maintenance technicians or operators can be given information about the shared characteristics or root causes of stoppages where that alarm sequence appeared in the past. This distils down the information that the technician is presented with, rather than having to deal with the high volume of individual alarms which can cause information overload.

1 Introduction

Operations and maintenance (O&M) costs for wind turbines can account for 18-30% of the cost of generation of wind power [1–4]. This is, in part, due to the highly irregular loads they experience from varied and turbulent wind conditions, so components can undergo high stress throughout their lifetime when compared with other rotating machines [5]. In addition, modern wind farms have large numbers of turbines, often deployed in remote or even offshore locations, making remote monitoring essential [6].

In the case of offshore turbines, access for maintenance or spot checks can be very expensive and weather dependent [7]. The transport cost for a maintenance crew to an offshore farm can range from €85/hr on relatively small vessels which are highly dependent on calm seas, to over €400/hr for larger vessels and helicopters [8]. Because of this, it is important to have robust control and monitoring systems which can detect faults and notify technicians, without raising false alarms. Furthermore, if a specific type of fault and an estimated time to failure can be predicted in advance, preventative measures can be taken to either avoid the fault or schedule maintenance at an optimal time, e.g. during good weather or during other maintenance activities to minimise the offshore trips needed. Hence, the offshore wind industry is driving maintenance from responsive or schedule-based activities to a more proactive and predictive strategy, with the need for strong remote monitoring capabilities [9].

Using machine learning and other data-driven analysis techniques on wind turbine Supervisory Control and Acquisition (SCADA) data to detect, diagnose and predict faults and turbine downtime is one way to ahcieve these goals [10–14]. A recent detailed review of all these methods can be found in [6], where the authors note that although there have been some successes, there is still work to be done before SCADA-based monitoring solutions can be commercially viable. To use these data-driven methods effectively, the data must be correctly labelled; i.e., the periods when the turbine was in faulty operation, or periods when a fault was developing, must be established [10]. As well as this, the root cause of the fault and its failure mode are important to characterise the fault in terms of severity and possible down time. Ways of identifying periods of faulty operation include cross-referencing with maintenance logs and using turbine alarm or availability data [15, 16]. However, this can be tedious and cumbersome; maintenance logs are not always standardised across turbines or even maintenance events, and may be stored as written documents rather than in a failure database [17]. Turbine alarms, on the other hand, are a promising way of identifying faulty periods, but often their effectiveness is reduced by the sheer volume of alarms generated.

In the following section, we give an overview of wind turbine alarms and why they can be hard to interpret, as well as work done in the area to improve their usefulness. We then give the motivation for our own work and the research objectives that this work hopes to achieve. In section 3, we outline the data used in this study. In section 4, we give an overview of our proposed methodology and apply it to an operating wind farm. In section 5, we discuss the effectiveness of the results achieved, and in section 6 give our conclusions and discuss future applications of the work.

2 Background

Turbine alarm systems vary widely between manufacturers, but generally share the same broad functionality. At their highest level, alarms (also called *events* or *statuses* by some manufacturers) are generated when the turbine operating state changes. There are usually different types of alarms depending on their severity:

- *Information alarms* are generally to communicate changes in certain operating conditions, e.g. when the wind speed is too low for generation, or a manual switch has been engaged.
- *Warning alarms*, on the other hand, are generated when the control system detects operating conditions or control variables that come close to exceeding certain thresholds.
- *Fault alarms* are generated when these thresholds are exceeded.

Note that the term "alarm" itself is sometimes used to refer to fault alarms exclusively by some original equipment manufacturers (OEMs), with information messages and warnings used to refer to the other types. In this research, we exclusively use the terms "information alarms", "warning alarms" and "fault alarms" as described

Table 1 Sample Alarms Data

Turbine	t_s	t_e	Code	Description	Category	Type
8	2015/06/01 17:13:38	2015/06/01 17:25:05	a_{41}	Storm shutdown - wind speed too high	Weather	Information
2	2015/06/02 10:19:05	2015/06/02 10:19:05	a_{124}	Motor Fuse Protection Warning	Generator	Warning
4	2015/06/03 07:56:14	2015/06/03 07:57:32	a_{91}	Low wind speed cut out	Weather	Information
10	2015/11/04 08:38:27	2015/11/04 08:38:37	a_{81}	Freq. Converter Line CCU current Fault	Freq. Conv.	Fault
6	2016/02/02 15:05:38	2016/02/02 15:11:01	a_{51}	Manual emergency stop initiated	User	Critical Fault
9	2016/02/01 01:26:46	2016/02/01 01:26:54	a_{22}	Normal Operation	No Fault	Information

above, whereas "alarms" by itself refers to any or all of the three in general.

Turbine alarms can occur in very high volumes. In [18], the authors found that the quantity of alarms raised is generally too high for operators to effectively manage. This is even higher during fault periods, when alarms occur in large and complex "showers", making failure detection, location and diagnosis difficult. By performing probability and time-based analyses, alarms which regularly appear together or trigger one another can be identified, and from this it is possible to identify particular failure modes related to alarm sequences.

In [19], the authors acknowledge the problems associated with the volume of alarms being generated by wind farms. They aimed to reduce this by using pattern recognition techniques based on artificial neural networks (ANNs) to identify alarm patterns which occurred during particular fault events. The authors found that by processing any alarm occurrences through the ANN model, faults could be flagged without the need for an operator to try and analyse the alarms themselves, thereby reducing the number of alarms that operators must work with. However, they noted that further work is needed to improve the accuracy of this method.

Mapping turbine alarms to a standardised taxonomy can help build a better picture of how faults propagate through various systems and understand the root cause of turbine stoppages [8, 20]. Taxonomy in this instance refers to how parts of the turbine are labelled and broken down into sub-systems (e.g. power module; rotor & blades; or nacelle) and assemblies within these sub-systems (e.g. the pitch system or blade bearings within the rotor & blades subsystem) in a standardised, OEM-agnostic fashion. Examples include the RDS-PP standard developed by VGB PowerTech e.V. [21], or ReliaWind-recommended taxonomy [22]. In [20], an extension to the ReliaWind taxonomy is proposed to standardise failure and reliability reporting across OEMs and turbine models. In [23], the authors show that by mapping turbine alarms to this taxonomy and performing a similar probability analysis to that of [18], alarms which are directly related to faults can be identified and the propagation of a fault from root cause to failure mode can be manually established.

Mapping the alarms to a taxonomy, or even in some cases decoding the meaning of some ambiguous alarms, can be difficult without intimate knowledge of the turbine control system. This "expert knowledge" is usually obtained by maintenance technicians or operators through training and experience. In the absence of such training or experience, good quality documentation and some basic knowledge of wind turbine technology can provide some of the information, but issues with data access and availability can limit the effectiveness of such an approach. This can make it difficult to know if some alarms are relevant to faults in a sub-system or assembly, or attributed to *reactions* to this fault.

In the case of faults in the pitch system, for example, if a pitch motor fault is detected, there are several contingency measures in place that kick in to avoid emergency situations. These include emergency brakes to stop the blades turning in case of a storm, and backup batteries in case power supply to the turbine is interrupted. Hence, if a pitch motor fault occurs, a number of alarms are generated to give information about the status of the auxiliary systems, or even faults in these auxiliary systems themselves, along with alarms related to the original fault. Without access to quality documentation and an intimate knowledge of the particular alarm system, it can, in some cases, be hard to decide which alarms are attributable to the original pitch motor fault, and which are to do with the auxiliary systems.

Furthermore, the presence of a turbine alarm is not always indicative of a fault, with the associated turbine shut-down intended to avoid damage taking place [20]. Tavner explains that turbine "failure rates" often cited in literature can really be regarded as "stoppage rates" [8]. These are usually the result of the turbine controller detecting an operational condition outside of acceptable bounds, such as over-speed, over-temperature or control problem, and generating a fault alarm. In the majority of cases, these stoppages result in an automatic or manual remote reset. However, historical stoppages >24 hours may be indicative that a manual inspection or repair was required as a result of some form of damage taking place.

If less severe stoppages of a shorter length of time are occurring frequently, they can be indicative of a wider problem on the turbine and can themselves contribute to reduced availability. However, because shorter stoppages are often resolved by a simple turbine reset, they can be overlooked by operators who may feel the effort required to analyse the density of alarms generated in such events is outweighed by the actual impact on turbine availability that any single short stoppage incurs.

In this paper, we aim to reduce the burden of analysis related to alarm showers which occur during turbine stoppages, however severe the reason for, or length of, the stoppage. This is achieved through two broad objectives:

- Identify "batches" of alarm sequences related to a particular turbine assembly which occur during stoppages
- Automatically group batches which contain similar alarm sequences together through the use of clustering techniques

In this way, each stoppage can be attributed to a specific type of sequence with its associated characteristics, rather than a large number of individual alarms which must be analysed. This reduces the burden of analysis for technicians as stoppages related to specific alarm sequences can be investigated for shared characteristics. This would allow information such as the probable root cause of the stoppage to be instantly known whenever such an alarm sequence reappears in future.

3 Description of Data

The data used in this study comes from an Irish wind farm with eleven 2.5 MW Doubly Fed Induction Generator (DFIG) turbines. The study covers a period of eleven months from June 2015 to April 2016. There were 118 days across all turbines where a maintenance team was on-site during this period. 56 of these days were due to 35 individual fault instances on the turbines which could not be fixed or diagnosed remotely. The remaining 62 days were due to scheduled periodic maintenance or upgrade work. Stoppages which did not require a maintenance call-out, e.g. when the turbine went down due to a fault which could be corrected remotely, were not recorded by the operator. The data used was alarm data from the turbines' OEM alarm system. A sample of this data can be seen in table 1.

Each alarm has a start time and an end time, though some alarms (e.g. certain information or warning alarms) can be instantaneous. Each specific alarm has an associated code and description, as well as a category and type. The categories of alarms are assigned by the OEM and are related to their functional location on the turbine, e.g. "Pitch", "Brake" or "Frequency Converter". Other categories contain alarms which are in some way logically connected, e.g. "weather" for weather-related alarms or "user" which are alarms which have been started by the user. The alarm type, on the other hand, is one of: information, warning, fault, or critical fault. This is also assigned by the OEM. These correspond to the descriptions of general turbine

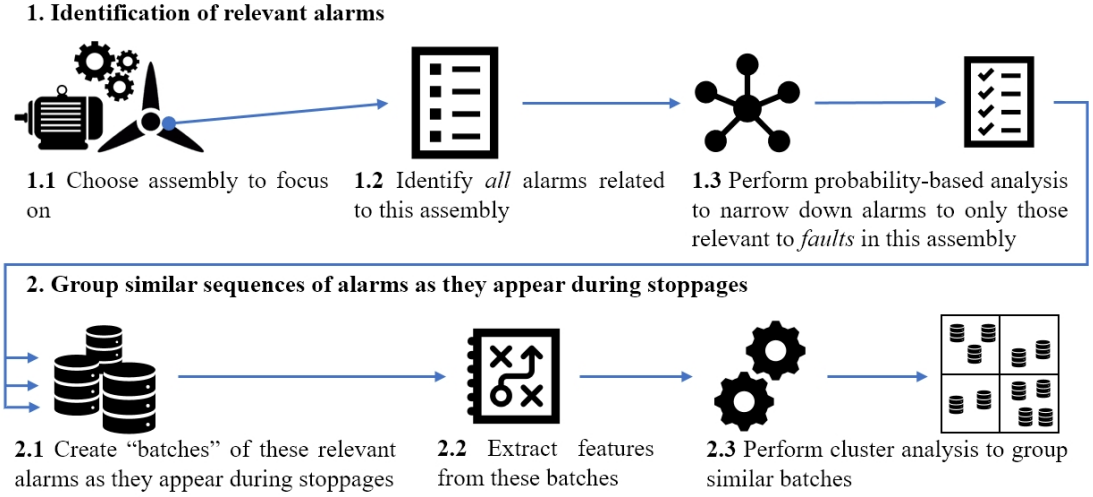

Fig. 1: Methodology overview

alarm systems given in section 2, with the exception that fault alarms are further split into "fault" and "critical fault". Both of these alarm types cause a turbine stoppage and require a reset. However, "critical faults" require a manual reset of the turbine as they generally warrant further investigation to make sure they are not part of a more serious issue.

Each alarm also gives the mean power output and wind speed which was seen over the course of the duration of the alarm. For both types of fault alarms, the power output is usually zero, or even negative where the turbine's systems are taking power from the grid to stay functional during downtime.

4 Methodology

The methodology developed in this research is split into two broad parts and summarised in figure 1. The first part focuses on identifying alarms relevant to potential faults which could occur in a particular assembly (e.g. the pitch system or frequency converter). We focus on a single assembly at a time in order to reduce the complexity of the analysis. The term "assembly" here refers to the definition as it appears in the ReliaWind taxonomy [8]. This taxonomy breaks down turbine parts into "sub-systems", e.g. the rotor or drive train, "assemblies" within these sub-systems, e.g. the pitch system within the rotor, "sub-assemblies" within the assemblies, e.g. the pitch drive, and individual "components" within these sub-assemblies, e.g. the pitch motors.

The second part of the methodology then focuses on identifying sequences, or "batches", of these alarms as they appeared during stoppages related to faults in the assembly, and using cluster analysis to group similar batches together. In this way, stoppages which share a similar sequence of relevant alarms can be grouped together. Further investigation can then be performed in order to identify a root cause for these stoppages. When a particular alarm sequence appears in future, the root cause will be known with minimal further analysis needed. The rest of this section is split into detailed sub-sections which correspond to each of the steps outlined in figure 1.

It should also be noted that, in this work, "alarm instance" refers to an individual alarm in the dataset, not to be confused with "alarm", which refers to that type of alarm (as opposed to a specific instance of it), or "alarm code", which refers to the code for that alarm. Alarm codes, and in some cases descriptions, have been changed for purposes of anonymity, and are referred to as a_1, a_2, a_3, etc. t_s refers to the start time of an alarm.

4.1 Identification of Relevant Alarms

4.1.1 Choose assembly to focus on: In order to reduce the complexity for analysis, a single turbine assembly at a time is analysed in this methodology, e.g. frequency converter, generator, pitch system, etc. Figure 2 shows the frequency of fault alarms according to their OEM-assigned category in the alarm system. These alarms are from all eleven turbines over the eleven-month period of the study, as detailed in section 3. As can be seen, the pitch system has the most frequent fault alarms. For this reason, we focus on this assembly in this study.

4.1.2 Identify all alarms related to this assembly: As stated in section 2, identifying alarms related to a specific assembly can be complicated by the volume of alarms and, in some cases, lack of clarity on their function or the relevant part of the taxonomy to which they belong. Hence, a probability-based analysis as described in [18] and [23] is performed which gives insights into groups of

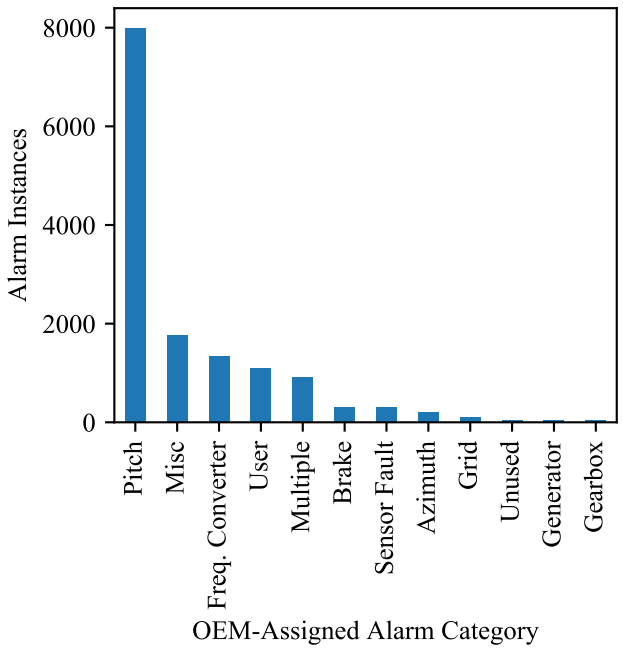

Fig. 2: Number of alarm instances with type "Fault" or "Critical Fault" by OEM-assigned category

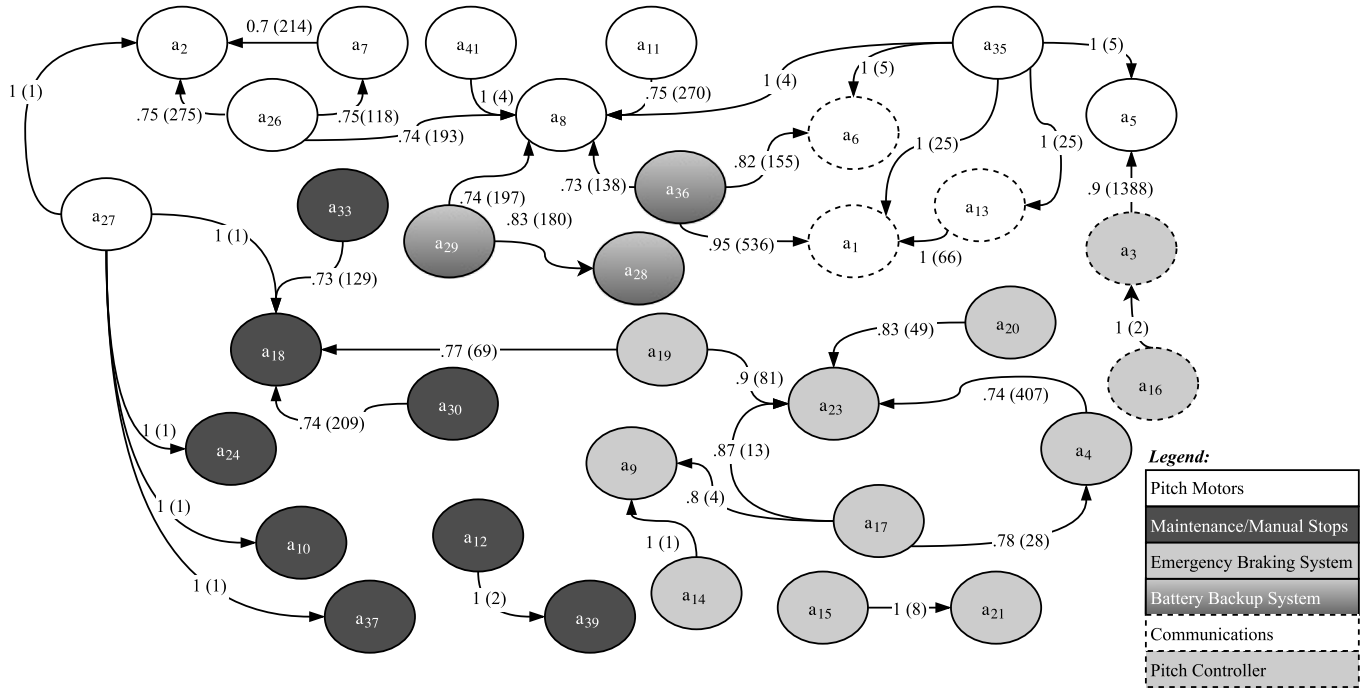

Fig. 3: Network diagram of pitch system alarm triggers, grouped according to the sub-assembly which they belong to. The arrows are labelled with the probability one alarm will trigger another, and the absolute number of times this happened shown in brackets

related alarms and which alarms trigger each other. This gives us an easy to interpret visual aid for what alarms could be important in determining periods of faulty operation related to a particular assembly.

Before doing the probability analysis, *all* possible alarms related to the assembly in question are identified. These include information, warning and fault (including critical fault) alarms related to the assembly itself as well as the auxiliary and support systems. If there are certain alarms where it is not clear to which system they belong to, they should be included anyway.

In this case, all alarms relating to the pitch system and its auxiliary support systems were included, for a total of 58 alarms, referred to here as L.

4.1.3 Perform probability-based analysis to narrow down alarms to only those relevant to faults in this assembly: The probability based analysis is performed as follows:

1. From the set of alarms L, all combinations of pairs of alarm codes are found, $\binom{L}{2}$.
2. For each pair of alarm codes a_1 and a_2, count the number of instances of alarm a_1 which have triggered one or more instances of a_2 and vice-versa.

An instance of a_1 is said to trigger an instance of a_2 if the following conditions are met:

$$t_{s_{a1}} <= t_{s_{a2}} \wedge t_{e_{a1}} >= t_{s_{a2}}$$

where $t_{s_{a1}}$, $t_{s_{a2}}$ and $t_{e_{a2}}$ are the start time of alarm instances a_1 and a_2, and the end time of instance a_1, respectively.

3. Calculate the probability that an instance of a_1 will trigger one or more a_2s, and vice-versa, where the probability of an instance of a_1 triggering one or more instances of a_2 is given as:

$$\Pr(a_{1trig}a_2) = |a_{1trig}a_2|/|a_1|$$

4. From here, the relationship between the two alarms will be determined as follows:

(a) If $\Pr(a_{1trig}a_2) >= 0.7$ and $\Pr(a_{2trig}a_1) >= 0.7$, then alarms a_1 & a_2 usually appear together
(b) If $\Pr(a_{1trig}a_2) <= 0.2$ and $\Pr(a_{2trig}a_1) <= 0.2$, a_1 & a_2 never or rarely appear together
(c) If $\Pr(a_{1trig}a_2) >= 0.7$ and $\Pr(a_{2trig}a_1) <= 0.2$, a_2 will usually be triggered whenever alarm a_1 appears; a_2 is a more general alarm
(d) If $\Pr(a_{1trig}a_2) <= 0.2$ and $\Pr(a_{2trig}a_1) >= 0.7$, a_1 will usually be triggered whenever alarm a_2 appears; a_1 is a more general alarm
(e) If none of the above, the two alarms are randomly or somewhat related

The results of this allow us to identify alarms related to different aspects of the chosen assembly, which will be analysed in the next step. We can exclude alarms related to specific auxiliary systems or manual intervention which are usually *reactions* to faults which have occurred in the assembly we are focusing on, or are simply not relevant. This step can be made easier by graphically analysing the results of the probability analysis in a network diagram. We refer to the final set of k alarm codes obtained as A_r:

$$A_r = [a_1, \ldots, a_k]$$

In this case, the probability-based analysis was performed on the 58 alarms identified in the previous step. A network diagram showing the relationships between these alarms was then constructed, as seen in figure 3. An alarm with an arrow leading from it to another alarm indicates that it usually triggers the other alarm. The numbers labelled along the arrows show the probability of one alarm triggering another, with the absolute number of times the alarm was triggered shown in brackets. Alarms which were not shown to generally trigger other alarms were left out of this diagram. This allows us to attribute alarms to various sub-assemblies or functions within the relevant assembly, where it is not clear from the documentation. As can be seen, there are a number of different "groups" of alarms, related to different sub-assemblies within the pitch system.

The alarms to be analysed in the next step were selected according to the following criteria:

- The alarm causes the wind turbine to stop generating
- The alarm is not related to a reaction to another alarm, e.g. safety-chain or maintenance-related alarms

• There were enough instances (> 25) of the alarm for it to be analysed effectively

Alarms related to the battery backup and emergency braking systems, and alarms related to manual control or maintenance were excluded, as well as alarms which do not cause the turbine to stop generating (e.g. information messages related to system tests). Of the remaining alarms, only those with enough instances for useful analysis were included, i.e. >25 instances. This number was found heuristically by iterating through this methodology to find a good balance of including relevant alarms without introducing too much noise caused by very rarely occurring alarms. This left us with a set of 30 alarms relevant to faults in the pitch system.

Alarms which represent the same fault, but, for example, on a separate turbine blade axis, were given the same shared alarm code. This was to ensure that alarm sequences along different axes would be grouped together as the same type of fault. For example, alarm codes a_{18}, a_{19} and a_{20} represent 'Pitch Control Deviation on Axis x fault', where x is 1, 2 or 3, respectively. All these alarm codes have been renamed a_{18}, to group them together as one. There were 12 of these "duplicate" alarms, to give us a final set of 18 relevant alarms for further analysis:

$$A_r = [a_1, a_2, \ldots, a_{18}]$$

4.2 *Group similar sequences of alarms*

4.2.1 Create "batches" of relevant alarm sequences: Before clustering, "batches" of fault alarm sequences which occur during stoppages must be identified.

The first step is to identify the alarm code that signifies the turbine returning to normal operation. This is usually an information alarm to communicate that the turbine has been brought back on-line after a fault alarm-related stoppage. In this case we refer to this code as a_n. Its associated description was "returning to normal operation". Once this has been found, the next step is to create "batches" of alarm sequences associated with each turbine, i.e. the alarm instances in each batch must all belong to the same turbine.

Each batch is created as follows. For the purposes of this description, "A_r alarms" refer to alarm instances whose code is one of the codes in A_r.

1. Find the earliest occurring A_r alarm in the data. Store its t_s as t_{start}
2. Find the next earliest occurring a_n alarm in the data. Store its t_s as t_{end}
3. Create a batch of A_r alarms with:

$$t_{start} \geq t_s < t_{end}$$

4. Select the next earliest occurring A_r alarm in the data. Store its t_s as t_{start}
5. Repeat steps 2-4 until no more A_r alarms in the data.

In our case, the results of this were a total of 456 batches of alarm sequences representing 456 individual stoppages across all 11 turbines in the 12 months of data. A typical example of a batch can be seen in table 2. It should be noted here that the final a_n alarm itself is not included in the batch at the analysis stage, but is provided when displaying batches so as to see how long the stoppage lasted. As can be seen, there are a mixture of alarms which occur individually and simultaneously (sharing common t_ss) to give a total number of 7 alarm instances with with four different t_ss.

Table 2 Example of a batch of alarm sequences (all alarms belong to same turbine)

t_s	Code	Description
24/12/2015 09:05:40	a_1	Blade angle asymmetry
	a_2	Pitch thyristor fault
24/12/2015 09:05:52	a_4	Blade braking time too high
24/12/2015 09:06:22	a_1	Blade angle asymmetry
	a_3	Pitch control deviation
	a_5	Pitch malfunction 2 or 3 blade
24/12/2015 09:06:57	a_5	Pitch malfunction 2 or 3 blade

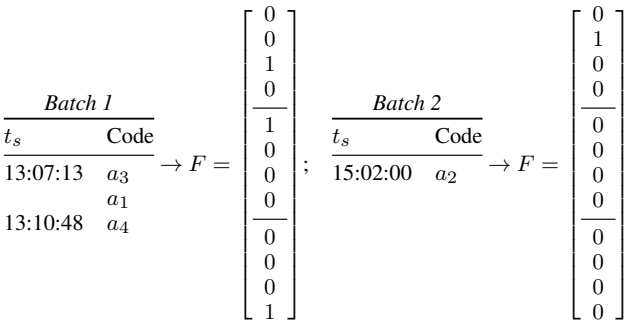

Fig. 4: Simplified examples of F_1 construction

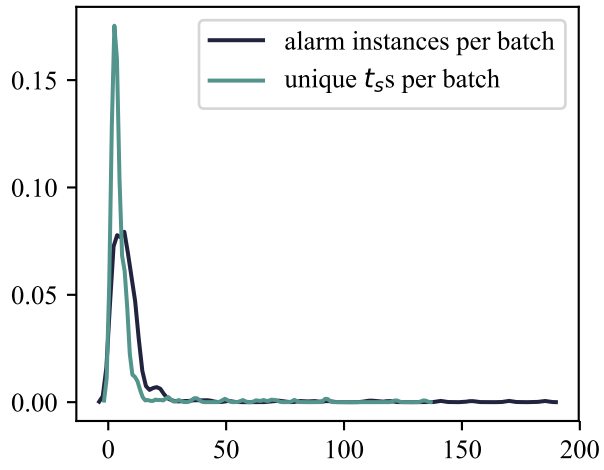

Fig. 5: Distribution of number of alarm instances and unique t_ss in each batch

4.2.2 Extract features from these batches: In order for the clustering to be effective, useful features from the alarm sequences must be extracted. Three separate ways of extracting feature vectors for each sample, F_1, F_2 and F_3, are explored.

F_1 - Base Case

The first feature extraction method was based solely on the order the alarms appeared in each batch. Batches can have a varying number of alarms, but in order to stop outlier batches with a disproportionately large number of alarms influencing the clustering algorithm, only batches with up to a certain maximum number of alarm instances, m_a, are included.

The feature vector for a batch, F_1, consists of a vector of 0s of length $k * m_a$, with a 1 being placed in the relevant location indicating the presence of an alarm:

$$F_1 = [f_1^1, f_2^1, \ldots, f_k^1, \ldots, f_1^{m_a}, f_2^{m_a}, \ldots, f_k^{m_a}]^T$$

A simplified example can be seen in figure 4, where there are four possible alarm codes, and a maximum of three alarm instances, i.e. $k = 4$, $m_a = 3$ and $A_r = [a_1, a_2, a_3, a_4]$. In batch 1, the first alarm is a_3, so a 1 is placed at f_3^1. The second alarm is a_1, so a 1 is placed at f_1^2. The third and final alarm is a_4, so a 1 is placed at f_4^3. In batch 2, there is only one alarm, a_2, so a 1 is placed at f_2^1. Note that the horizontal lines in the vector here are just to make the example easier to interpret.

Batch 1

t_s	Code
13:08:02	a_1
	a_2
	a_4
13:08:50	a_3
13:12:10	a_1
	a_3

$\rightarrow F =$ $\begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ \hline 0 \\ 0 \\ 1 \\ 0 \\ \hline 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$;

Batch 2

t_s	Code
15:02:30	a_1
	a_2

$\rightarrow F =$ $\begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ \hline 0 \\ 0 \\ 0 \\ 0 \\ \hline 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$

Fig. 6: Simplified examples of F_2 construction

Batch 1

t_s	Code
13:08:02	a_1
	a_2
	a_4
13:08:50	a_3
13:12:10	a_1
	a_3
13:15:10	a_n

$\rightarrow F =$ $\begin{bmatrix} 100 \\ 100 \\ 0 \\ 100 \\ 48 \\ \hline 0 \\ 0 \\ 100 \\ 0 \\ 140 \\ \hline 100 \\ 0 \\ 100 \\ 0 \\ 150 \end{bmatrix}$;

Batch 2

t_s	Code
15:02:30	a_1
	a_2
15:04:35	a_n

$\rightarrow F =$ $\begin{bmatrix} \text{X} \\ \text{X} \\ 0 \\ 0 \\ 125 \\ \hline 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \hline 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$

Fig. 7: Simplified examples of F_3 construction

Figure 5 shows the distribution of number of alarm instances and unique t_ss per batch in the dataset used in this study. As can be seen, over 90% of batches had between 1 and 10 alarm instances, so in this case $m_a = 20$ was selected as the maximum number of alarm instances. This led to 425 batches with an average of 6.75 alarms per batch. With $k = 18$ possible alarms, the length of each feature vector F was $18 * 20 = 360$.

***F_2* - Incorporating simultaneous start times**

The batch in table 2 has a number of alarms occurring simultaneously. This is a similar case for many batches, so it was decided to extract a feature set that takes this into account by grouping alarms according to their t_s. To protect from the influence of outliers, only batches with up to a certain maximum number of t_s, m_t, were included. This is similar to F_1, where only batches with up to a certain number of unique alarms were included.

The feature vector for a sample, F_2, once again consisted of a vector of 0s, this time of length $k * m_t$:

$$F_2 = [f_1^1, f_2^1, \ldots, f_k^1, \ldots, f_1^{m_t}, f_2^{m_t}, \ldots, f_k^{m_t}]^T$$

A simplified example is shown in figure 6, using $k = 4$, $m_t = 3$ and $A_r = [a_1, a_2, a_3, a_4]$. In the first batch, there are three alarms occurring at the first t_s (13:08:02), a_1, a_2 and a_4, so 1s are placed at f_1^1, f_2^1 and f_4^1. There is only one alarm, a_3 at the next t_s, so an alarm is placed at f_3^2. Two alarms, a_1 and a_3 occur at the final t_s, so 1s are placed at f_1^3 and f_3^3

As seen in figure 5, over 90% of batches have between 1 and 10 unique t_ss, so m_t was set to 10. This translated to 417 batches, with a mean of 6.57 alarms spread across 3.84 t_ss in each batch. Each feature vector was $18 * 10 = 180$ long.

***F_3* - Incorporating the time between each t_s**

The final feature extraction method expands on the previous method by incorporating the time between each t_s, representing how long the alarms at that t_s persisted before other alarms were triggered. It does this by making two slight changes to F_2. First, the time in seconds between each $t_s$is added as an extra feature at the end of each group of k alarms seen in F_2. In the case of the last alarm, the time between its t_s and the t_s of the "returning to normal operation" alarm for that batch, a_n, is used. This means the final length of the vector is $(k + 1) * m_t$.

Because the new features can be $\gg 1$, there is a chance the clusters could be heavily biased towards grouping batches with similar numbers of t_s, and the time between these t_ss, without taking into account the actual alarm codes themselves. Hence, the second change is that different values can be substituted for 1, such as 100, 1000, etc.

An example is provided in figure 7. This is identical to the example in the last section, but with the extra features added. Note that in this example $X = 100$, so 1s are replaced with 100s. In the first batch, the first "extra" feature is 48, signifying the time difference in seconds between the first t_s (13:08:02) and the second t_s (13:08:50), so this is placed at position f_5^1. The fourth and final t_s is 13:12:10, and the t_s of a_n is 13:15:10. This is 150s after the final t_s, so 150 is placed at position F_5^4. Note that the a_n alarms are included here only to show where the final "extra" feature comes from; they aren't included in batches during cluster analysis.

As before, we use batches with between 1 and 10 unique t_ss, for a total of 417 batches. Each feature vector this time was $(18 + 1) * 10 = 190$ long.

4.2.3 Perform cluster analysis: Identifying patterns in high-dimensional data with no "ground truth" to learn from is an unsupervised learning problem [24]. Cluster analysis is a powerful unsupervised learning technique that is used to identify patterns in samples of data and group samples with similar patterns together, so is ideally suited to this problem [25]. The main goal of clustering is to group a collection of objects into separate subsets or "clusters", so that the objects in each cluster are more similar to each other than those in other clusters.

Agglomerative clustering is a type of hierarchical, or tree-based, clustering which is well suited to data that has a large number of clusters. In this case, there may be many different alarm sequences so this is an appropriate technique to use. It starts by assigning every individual sample into its own unique cluster. It then looks at the pairwise similarity between all the clusters, and merges the two which are most similar to each other. It repeats this process until some specified number of clusters remain. The result of this is a binary tree linking each sample into one of a number of clusters [25]. In this work, the Euclidean distance between the centre of each cluster is used as a similarity metric.

Density-based spatial clustering of applications with noise (DBSCAN), is another powerful clustering technique that does not need many parameters and automatically decides on an optimal number of clusters. DBSCAN views clusters as areas of high density (i.e. many samples in close proximity to each other) separated by areas of low density. It does this by first assigning some samples as "core" samples. These are defined as samples which have a certain minimum number of "neighbours", with neighbours being defined as samples within some minimum amount of distance to them. Clusters are built by recursively selecting a core sample, finding all of its neighbours which are core samples, finding all of their neighbour core samples, and repeating until there are no further neighbour core samples within that cluster. All other samples which are not core samples (i.e. don't meet the minimum number of neighbours), but which are themselves neighbours of a core sample, are assigned to that core sample's cluster. Any samples which are not in the neighbourhood of any core sample are classed as outliers, and not assigned a cluster. In this way, DBSCAN decides on its own optimal number of clusters[26]. Because DBSCAN automatically decides on an optimal number of clusters, it was decided to compare this with agglomerative clustering in this work.

Clustering performance was evaluated in two ways. First, the silhouette coefficient was used as a measure of how well defined the clusters are. The silhouette coefficient is defined as follows:

$$s = \frac{b - a}{max(a, b)}$$

Table 3 Results Summary

Feature Set - Algo.	No. Clusters	Avg. Sil.	$\% > .9$	Accurate Sil.?
1 - Agg	20	.2	16	Y
1 - DBSCAN	13	1	27.3	Y
2 - Agg	20	.39	3.8	Y
2 - DBSCAN	15	1	45.1	Y
3 - Agg ($X = 1$)	8	.82	91.4	N
3 - DBSCAN ($X = 1$)	7	.93	41.2	N

where a is the mean distance between a sample and all other samples in the same cluster, and b is the mean distance between a sample and all other points in the next nearest cluster. The silhouette coefficient takes a value between -1 and 1, with 1 meaning the point is far away from its neighbouring cluster, 0 meaning it's on the boundary, and -1 meaning the point has possibly been misclassified. The silhouette coefficient is hence a score given to every sample in a cluster and is evaluated graphically, as will be seen in section 5.

The silhouette coefficient is a good indication of how well the clustering is performing, but only if the features that have been extracted accurately represent the underlying data. For this reason, in some cases, a manual inspection of the clusters was performed to ensure that good/bad silhouette scores translated to effective clustering for this specific use case. The manual inspection involved selecting 2-3 samples from each cluster and checking if the alarm sequences in each sample were similar to each other if there was a high silhouette score, or dissimilar for a low silhouette score.

The agglomerative clustering and DBSCAN algorithms were applied to the three different feature sets extracted in the previous step, F_1, F_2 and F_3, with F_3 being trained with various different values of X; $X \in \{1, 10, 100, 1000\}$. Since agglomerative clustering takes a specific number of clusters as an input, it is normally trained several times with a number of different clusters to find the optimal number. Here, the analysis was carried out for between 2 and 20 clusters, with the optimal number of clusters being selected according to the one with the highest silhouette coefficient.

Once the optimum number of clusters has been found for agglomerative clustering, it is evaluated against DBSCAN. The final evaluation is performed again using the silhouette score, with the effectiveness of the score being checked via manual inspection. The most effective method is decided heuristically, e.g. if agglomerative clustering manages to group 40% of clusters with reasonable accuracy, but DBSCAN classifies 30% of clusters with perfect accuracy, then DBSCAN in that case would be a better choice. The results of this can be found in the following section.

5 Clustering Results

As described in section 4.2.1 (step 2.1 from figure 1), 456 batches were created from the full set of data. Three sets of features, F_1, F_2 and F_3 were extracted from these batches, as described in section 4.2.2 (step 2.2 of figure 1). With m_a set to 20 and m_t set to 10, this meant there were 425 samples of F_1 and 417 samples of F_2 and F_3. The results of applying the clustering method described in section 4.2.3 (step 2.3 of figure 1) are discussed in this section.

In all cases, both silhouette and manual analyses were performed. A summary of the results can be seen in table 3. This table shows the name, no. of clusters, silhouette score, and % of samples which achieved a silhouette score of >0.9. The table also shows whether or not the silhouette score gave a good indication of accurate clustering, as determined from manual inspection. Note $X = 1$ is the only one included for feature set 3 as this was the best scoring value of X.

5.1 F_1 - *Base Case*

5.1.1 Agglomerative Clustering: Silhouette analysis was carried out for the agglomerative clustering with between 2 and 20 clusters. The silhouette analysis for the case of 3, 8 and 20 clusters can be seen in figure 8.

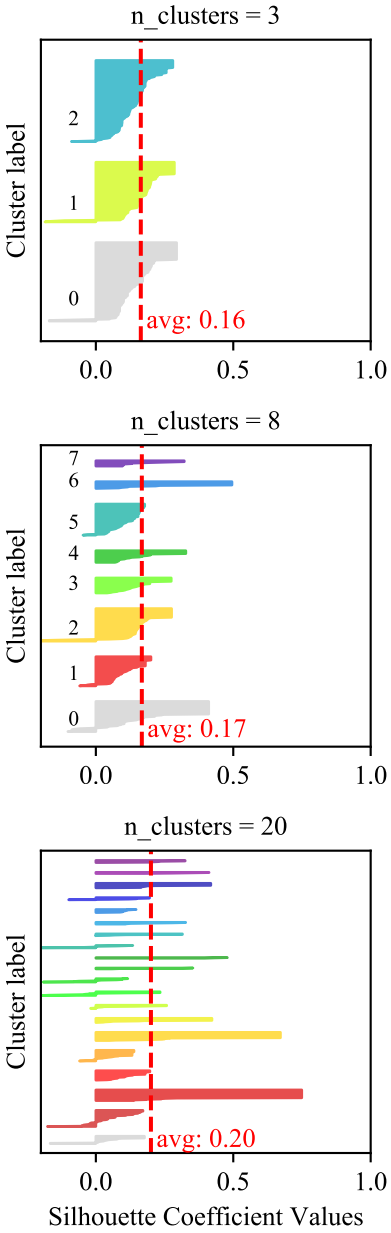

Fig. 8: Silhouette analysis for agglomerative clustering using feature set 1

In this figure, each cluster label represents the silhouette scores of every batch sample in that cluster, sorted in increasing order. This means that the thicker the silhouette plot for each cluster, the more samples there are in that cluster.

As can be seen, a higher number of clusters in agglomerative clustering performed better. The best average silhouette score was found on the maximum 20 clusters, with an average silhouette score of .2. However, there were big fluctuations in the silhouette scores of members within each cluster. Manual inspection confirmed that the clusters that scored well contained batches that had very similar alarm sequences, however the clusters that scored above 0.9 represented only 16% of samples fed into the algorithm.

5.1.2 DBSCAN: DBSCAN in this case classified 27.3% of samples into 13 different clusters. The average silhouette score across all clusters was 1. A manual inspection confirmed that the sequences of alarms in samples within each cluster were identical in nearly all cases.

Table 4 Samples within a high scoring cluster for DBSCAN using F_1

t_s	Code	Description
	Sample 1	
2016-05-05 13:08:42	a_6	Pitch controller comms fault
2016-05-05 13:08:55	a_5	Pitch malfunction 2 or 3 blade
	Sample 2	
2015-06-26 12:25:11	a_6	Pitch controller comms fault
	a_5	Pitch malfunction 2 or 3 blade

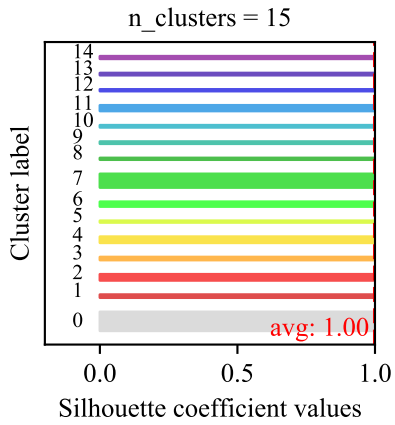

Fig. 9: Silhouette scores for DBSCAN using F_2 features

An important point to note is that because the F_1 features did not take into account whether some alarm instances appeared simultaneously, in a small number of cases there were different numbers of t_ss in each sample within a cluster. An example of this can be seen in table 4, showing two samples from the same cluster. In *Sample 1* the 2 alarm instances happen in sequence, whereas in *Sample 2*, they happen simultaneously. This can be relevant as the root cause related to the alarm sequence in both cases could possibly be different; in *Sample 1* there was a pitch malfunction in the blades (i.e. the pitch angles in all three blades were not equal), which was caused by a communication fault in the pitch controller. In *Sample 2* the two occurred simultaneously, which in cases with more complex alarm sequences could point to different root causes.

5.2 F_2 - *Incorporating Simultaneous Start Times*

5.2.1 Agglomerative Clustering: Here, the optimum number of clusters for agglomerative clustering was again found to be 20, with a silhouette score of 0.39. Only 3.8% of samples were clustered with a score above 0.9.

5.2.2 DBSCAN: DBSCAN once again performed much better than agglomerative clustering, with an average silhouette score of 1 across 15 clusters, as seen in figure 9. This represented 45.1% of samples fed into the algorithm. A manual investigation of the clusters revealed that not only were the alarm sequences in each sample within clusters identical, but each sample also had the same number of t_ss, i.e. the information about alarm instances that occurred simultaneously was preserved.

5.3 F_3 - *Incorporating the Time Between Each* t_s

The above analysis was repeated using the new time-based features for various values of X.

For this analysis, the feature array was normalised before clustering, to avoid large times between t_ss having a disproportionate impact.

Table 5 Example of alarm sequences found in cluster 2

t_s	Code	Description
2015/06/29 14:44:29	a_1	Blade angle asymmetry
	a_2	Pitch thyristor fault
2015/06/29 14:44:34	a_4	Blade braking time too high
2015/06/29 14:44:37	a_1	Blade angle asymmetry
	a_3	Pitch control deviation
	a_5	Pitch malfunction 2 or 3 blades
2015/06/29 14:44:38	a_3	Pitch control Deviation
	a_5	Pitch malfunction 2 or 3 blades
2015/06/29 14:52:10	a_n	*Returning to normal operation*

5.3.1 Agglomerative Clustering: The optimal number of clusters across all values of X was found to be 8. With 8 clusters, the highest average silhouette score was 0.85, for $X = 1$. However, a manual inspection of the clusters showed that the samples in each varied wildly. This was probably down to the fact that setting $X = 1$ means the clustering barely takes into account the actual alarms that were generated, and focuses almost solely on the times between each t_s in a batch, which could be much greater than 1. Even after normalisation, the average value of features representing these times between each t_s was 0.331, whereas the value of the features representing the presence of a particular alarm code (i.e. the features which are marked as "1" for $X = 1$) was 0.001.

For $X = 10$, $X = 100$ and $X = 1000$, silhouette scores were 0.82, 0.52, and 0.26, respectively. However, once again the batches in each cluster were quite different. With manual inspection, it was found that in batches with identical alarm sequences, there was a wide range of possible values for the time between each t_s, i.e. even though the sequences of alarm instances in two different batches could be identical, the time between these alarms could considerably vary. This could mean that effective clustering using these extra features was not possible.

5.3.2 DBSCAN: DBSCAN produced 7 clusters for all values of X, with silhouette scores of 0.93, 0.91, 0.8 and 0.43 for $X = 1$, $X = 10$, $X = 100$ and $X = 1000$, respectively. Once again manual inspection showed that there was wide variation in the samples within each cluster. This added further evidence to the fact that the extra features created for feature set 3 were not suitable for effective clustering.

5.4 *Analysis of Results*

Based on the above results, DBSCAN performed on the F_2 features yielded the best results. This resulted in 15 clusters of batches, with each batch containing an identical sequence of alarms. 45.1% of batches fed into the clustering algorithm were successfully assigned a group, which represented 41% of the 456 total batches analysed in the study. These correctly clustered batches together represented over 134 hours of downtime on the turbine, with each stoppage lasting an average of just below 15 minutes.

A sample of a batch from cluster 2 can be seen in table 5, showing the progression of a fault in the pitch system. Once again, the alarm a_n here is just provided to show how long the stoppage lasted in total. First, a fault in the thyristor of one of the pitch motor circuits is detected, which simultaneously causes asymmetry in the pitch angles across the three blades. Because of this, the turbine isn't braking quickly enough, which sets off the a_4 alarm, as well as blade angle asymmetry alarms for the other blades, a pitch control deviation alarm and a more "general" alarm showing a pitch malfunction across more than one blade. The other batches in this cluster showed the exact same alarm sequence, including which alarms occurred simultaneously.

Overall, these results show that a large proportion of the alarm sequences which occur during individual stoppages associated with the pitch system can be accurately sorted into a number of distinct groups. The implications of this will be discussed in the following section.

6 Conclusion

This work focused on attempting to sort similar sequences of alarms as they occurred during wind turbine stoppages into several distinct groups, with the aim of reducing the burden of analysis on turbine operators when high volume alarm showers are generated. The alarms generated during 456 different stoppages were analysed. Sequences of alarms as they occurred during each stoppage were identified, with each "batch" of alarm sequences being associated with a particular stoppage. Three different sets of features representing the alarms in each batch were extracted, and clustering techniques applied with the aim of grouping similar batches together. The first feature set looked solely at the order the alarms appeared in. The next set took into account whether or not some alarms occurred simultaneously. The third feature set took into account the time between the alarms in each batch. Two different clustering techniques, agglomerative clustering and DBSCAN, were applied to these three feature sets, and the results of each compared.

The results for the first feature set showed promise, with DBSCAN managing to accurately cluster 27.4% of samples. A drawback was that the samples within each cluster did not take into account whether some alarm instances occurred simultaneously or not. Agglomerative clustering in this case showed poor results. The best results occurred on the second feature set using DBSCAN; 45.1% of batches were accurately sorted into fifteen distinct clusters. In this case, whether or not some alarms occurred simultaneously was consistent within batches. Agglomerative clustering once again did not perform as well as hoped. The third feature set showed poor results all round, possibly due to there being too much variance of possible values for the time between alarm instances.

Based on these results, it is indeed possible to usefully group together similar sequences of alarm instances into distinct clusters. This means that the burden of analysis for turbine operators during stoppages can be reduced. If a stoppage occurs during live operation, and the resulting sequence of alarms can be attributed to a previously identified group of similar alarm sequences which occurred during past stoppages, the operator can be given information about the shared characteristics of these stoppages rather than seeing a cascade of individual alarms which need to be analysed. This information can be related to what corrective action, if any, was generally taken in the past, the severity of the fault and duration of associated down time, the root cause or other information to help diagnose the fault, whether the stoppage was controller-related, or others. As well as this, the frequency of particular alarm sequences can be tracked, which can give more information and context than simply tracking the frequency of individual alarms.

The natural extension to this work involves investigating the different types of stoppages to identify their shared characteristics. Once these characteristics have been identified, not only can they be used for diagnosing future faults and deciding on the appropriate course of action post-occurrence, but can also be used for predictive purposes. By overlaying the times that particular past alarm sequences occurred over historical SCADA data, machine learning models can be trained to find certain leading indicators in the operational data that this particular type of alarm sequence may be imminent. As the live data then shows some of these indicators, operators can be given advance warning of a specific type of fault, along with an estimated window of when the fault will occur. With knowledge of the set of shared characteristics that this stoppage is likely to have, an appropriate course of action can be decided upon in advance, and preventative, rather than corrective, action can be taken.

Acknowledgement

This research was funded by Science Foundation Ireland (SFI) Centre MaREI - Centre for Marine and Renewable Energy (12/RC/2302).

7 References

1 Krohn, S., Morthorst, P.E., Awerbuch, S.: 'The Economics of Wind Energy'. (European Wind Energy Association, 2009)
2 Tchakoua, P., Wamkeue, R., Ouhrouche, M., Slaoui.Hasnaoui, F., Tameghe, T., Ekemb, G.: 'Wind Turbine Condition Monitoring: State-of-the-Art Review, New Trends, and Future Challenges', *Energies*, 2014, **7**, (4), pp. 2595–2630
3 Walford, C.a.: 'Wind turbine reliability: understanding and minimizing wind turbine operation and maintenance costs', *Energy*, 2006, , (March), pp. SAND2006–1100
4 Peters, V.A., Ogilvie, A.B., Bond, C.R. 'Continuous Reliability Enhancement for Wind (CREW) Database : Wind Plant Reliability Benchmark'. (, 2012. September
5 Kusiak, A., Verma, A.: 'A Data-Mining Approach to Monitoring Wind Turbines', *IEEE Transactions on Sustainable Energy*, 2012, **3**, (1), pp. 150–157
6 Tautz.Weinert, J., Watson, S.J.: 'Using SCADA data for wind turbine condition monitoring - a review', *IET Renewable Power Generation*, 2017, **11**, (4), pp. 382–394
7 Faulstich, S., Hahn, B., Tavner, P.J.: 'Wind turbine downtime and its importance for offshore deployment', *Wind Energy*, 2011, **14**, (3), pp. 327–337
8 Peter Tavner: 'Offshore Wind Turbines: Reliability, availability and maintenance'. (Institution of Engineering and Technology, 2012)
9 Yang, W., Tavner, P.J., Crabtree, C.J., Feng, Y., Qiu, Y.: 'Wind turbine condition monitoring: Technical and commercial challenges', *Wind Energy*, 2014, **17**, (5), pp. 673–693
10 Leahy, K., Hu, R.L., Konstantakopoulos, I.C., Spanos, C.J., Agogino, A.M.: 'Diagnosing Wind Turbine Faults using Machine Learning Techniques Applied to Operational Data', *2016 IEEE International Conference on Prognostics and Health Management (ICPHM)*, 2016, pp. 1–8
11 Hu, R.L., Leahy, K., Konstantakopoulos, I.C., Auslander, D.M., Spanos, C.J., Agogino, A.M. 'Using Domain Knowledge Features for Wind Turbine Diagnostics'. In: 2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA). (IEEE, 2016. pp. 300–307
12 Kusiak, A., Li, W.: 'The prediction and diagnosis of wind turbine faults', *Renewable Energy*, 2011, **36**, (1), pp. 16–23
13 Kusiak, A., Verma, A.: 'A data-driven approach for monitoring blade pitch faults in wind turbines', *IEEE Transactions on Sustainable Energy*, 2011, **2**, (1), pp. 87–96
14 Yang, W., Court, R., Jiang, J.: 'Wind turbine condition monitoring by the approach of SCADA data analysis', *Renewable Energy*, 2013, **53**, pp. 365–376
15 Godwin, J.L., Matthews, P.: 'Classification and Detection of Wind Turbine Pitch Faults Through SCADA Data Analysis', *International Journal of Prognostics and Health Management*, 2013, **4**, pp. 11
16 Chen, B., Matthews, P.C., Tavner, P.J.: 'Wind turbine pitch faults prognosis using a-priori knowledge-based ANFIS', *Expert Systems with Applications*, 2013, **40**,

(17), pp. 6863–6876

17 Kaidis, C., Uzunoglu, B., Amoiralis, F.: 'Wind turbine reliability estimation for different assemblies and failure severity categories', *IET Renewable Power Generation*, 2015, **9**, (8), pp. 892–899

18 Qiu, Y., Feng, Y., Tavner, P., Richardson, P., Erdos, G., Chen, B.: 'Wind turbine SCADA alarm analysis for improving reliability', *Wind Energy*, 2012, **15**, (8), pp. 951–966

19 Chen, B., Qiu, Y.N., Feng, Y., Tavner, P.J., Song, W.W.: 'Wind turbine SCADA alarm pattern recognition', *IET Conference on Renewable Power Generation (RPG 2011)*, 2011, pp. 363–368

20 Reder, M., Gonzalez, E., Melero, J.J.: 'Wind Turbine Failure Analysis - Targeting current problems in Failure Data Analysis', *Journal of Physics: Conference Series*, 2016, **753**, (072027)

21 PowerTech, V. 'RDS-PP âĂŞ Application Guideline; Part 32: Wind Power Plants'. (, 2014.

22 Wilkinson, M., Hendriks, B., Spinato, F., Delft, T.V.: 'Measuring Wind Turbine Reliability - Results of the Reliawind Project', *European Wind Energy Association Conference*, 2011, pp. 1 – 8

23 Gonzalez, E., Reder, M., Melero, J.J.: 'SCADA alarms processing for wind turbine component failure detection', *Journal of Physics: Conference Series*, 2016, **753**, (7), pp. 072019

24 James, G., Witten, D., Hastie, T., Tibshirani, R.: 'An Introduction to Statistical Learning'. vol. 103 of *Springer Texts in Statistics*. (New York, NY: Springer New York, 2013)

25 Hastie, T., Tibshirani, R., Friedman, J.: 'The Elements of Statistical Learning'. vol. 1 of *Springer Series in Statistics*. (New York, NY: Springer New York, 2009)

26 Ester, M., Kriegel, H.P., Sander, J., Xu, X. 'A density-based algorithm for discovering clusters in large spatial databases with noise.'. In: Knowledge Discovery and Data Mining. (, 1996. pp. 226—-231